

MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM		MMM	AAA	AAA	CCC	RRRRRRRRRRRR	000	000
MMM		MMM	AAA	AAA	CCC	RRRRRRRRRRRR	000	000
MMM		MMM	AAA	AAA	CCC	RRRRRRRRRRRR	000	000
MMM		MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM		MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM		MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM		MMM	AAA	AAA	CCC	RRR	RRR	000
MMM		MMM	AAA	AAA	CCC	RRR	RRR	000
MMM		MMM	AAA	AAA	CCC	RRR	RRR	000
MMM		MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000
MMM		MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000
MMM		MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000

LL	IIIIII	NN	NN	KK	KK	
LL	IIIIII	NN	NN	KK	KK	
LL	II	NN	NN	KK	KK	
LL	II	NN	NN	KK	KK	
LL	II	NNNN	NN	KK	KK	
LL	II	NNNN	NN	KK	KK	
LL	II	NN	NN	KKKKKK		
LL	II	NN	NN	KKKKKK		
LL	II	NN	NNNN	KK	KK	
LL	II	NN	NNNN	KK	KK	
LL	II	NN	NN	KK	KK
LL	II	NN	NN	KK	KK
LLLLLLLLLLLL	IIIIII	NN	NN	KK	KK
LLLLLLLLLLLL	IIIIII	NN	NN	KK	KK

LL	IIIIII	SSSSSSSS	
LL	IIIIII	SSSSSSSS	
LL	II	SS	
LL	II	SS	
LL	II	SS	
LL	II	SS	
LL	II	SSSSSS	
LL	II	SSSSSS	
LL	II		SS
LL	II		SS
LL	II		SS
LL	II		SS
LL	II		SS
LLLLLLLLLLLL	IIIIII	SSSSSSSS	
LLLLLLLLLLLL	IIIIII	SSSSSSSS	

(2)	57	declarations
(3)	100	.link -- process the .link directive
(4)	260	get_file_name accumulate file name
(5)	317	process_qual process link directive qualifiers
(7)	431	get_incl_list Get the module(s) in the list
(8)	500	insert_module Insert module name into list
(9)	553	build_lnk_rec Build a linker options record
(11)	649	mac\$wrt_lnkopt Write the linker options records to object

```
0000 1      .title mac$Link      link directive processor
0000 2      .ident 'V04-000'
0000 3      :
0000 4      :*****
0000 5      :*
0000 6      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      :*  ALL RIGHTS RESERVED.
0000 9      :*
0000 10     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     :*  TRANSFERRED.
0000 16     :*
0000 17     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     :*  CORPORATION.
0000 20     :*
0000 21     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     :*
0000 24     :*
0000 25     :*****
0000 26     :
0000 27     :++
0000 28     :
0000 29     : Facility:
0000 30     :
0000 31     :     VAX-11 Macro Assembler
0000 32     :
0000 33     : Abstract:
0000 34     :
0000 35     :     This module contains the routines required to handle the .LINK
0000 36     :     assembler directive. The nature of this directive is to allow
0000 37     :     the user to specify linker options within the object module
0000 38     :     produced by the assembler.
0000 39     :
0000 40     : Environment:
0000 41     :
0000 42     :     Native Mode, User Mode
0000 43     :
0000 44     : Author:
0000 45     :
0000 46     :     Michael T. Rhodes,      Creation Date: April, 1983
0000 47     :
0000 48     : Modified By:
0000 49     :
0000 50     :     V03-001 MTR0036      Michael T. Rhodes      16-Aug-1983
0000 51     :     Add abbreviated qualifier name synonyms and adjust CASE
0000 52     :     table dispatch address to accomodate the LNK$C_SHR object
0000 53     :     record type.
0000 54     :
0000 55     :--
```

```
0000 57      .sbttl  declarations
0000 58
0000 59  ::
0000 60  :: Macros:
0000 61  ::
0000 62      $macmsgdef      ; Define Macro-32's messages.
0000 63      $mac_ctlflgdef  ; Define the control flags.
0000 64      $mac_objcoddef  ; Define the object code.
0000 65      $mac_symlkdef   ; Define the symbol block offsets etc..
0000 66
0000 67  ::
0000 68  :: Equated Symbols:
0000 69  ::
0000000D 0000 70 cr = ^X0D      ; Carriage return token.
00000013 0000 71 lnk_blk_siz = ^X13 ; Size of the fixed portion of the linker op
00000004 0000 72 lnk_q_inclst = ^X4  ; Offset to include list address.
0000000C 0000 73 lnk_l_bytes = ^XC   ; Offset to #bytes required for include list
00000010 0000 74 lnk_l_states = ^X10 ; Offset to link directive state flags.
0000 75
0000 76
00000000 77      .psect  mac$rw_data, noexe, rd, wrt
0000 78  ::
0000 79  :: Linker option records list head.
0000 80  ::
0000 81  mac$qgq_lnkopt::
00000000' 0000 82      .address mac$qgq_lnkopt
00000000' 0004 83      .address mac$qgq_lnkopt
0000 84
00000000 85      .psect  mac$ro_data, noexe, nowrt, gbl, long
0000 86  ::
0000 87  :: Linker qualifier options table.
0000 88  ::
00000000 0000 89 insymp = 0
0000 90      $mac_insert_syx SH,      lnk$sc_sha
000C 91      $mac_insert_syx SE,      lnk$sc_obj
0018 92      $mac_insert_syx I,      lnk$sc_oli
0023 93      $mac_insert_syx L,      lnk$sc_olb
002E 94      $mac_insert_syx SHAREABLE, lnk$sc_sha
0041 95      $mac_insert_syx SELECTIVE_SEARCH, lnk$sc_obj
005B 96      $mac_insert_syx INCLUDE, lnk$sc_oli
006C 97      $mac_insert_syx LIBRARY, lnk$sc_olb,      lnk_qualifiers
007D 98
```

```

007D 100 .sbtll .link -- process the .link directive
007D 101 :++
007D 102 :
007D 103 : Functional Description:
007D 104 :
007D 105 : This routine is called to process the .LINK directive.
007D 106 : The valid syntax for this directive is as follows:
007D 107 :
007D 108 : .LINK 'filespec'[qualifier[=module or (module list)]],...
007D 109 :
007D 110 : The filespec within the delimiters is scanned (built into a
007D 111 : .ASCII string), then we scan looking for a .LINK directive qualifier.
007D 112 : If none are present, the default linker option record type (regardless
007D 113 : of the extension specified in the filespec) is OBJECT (which includes
007D 114 : symbol tables).
007D 115 :
007D 116 : Implicit Inputs:
007D 117 :
007D 118 : mac$ab_tmpbuf adr The address of the assembler's temporary buffer
007D 119 : used to accumulate the delimited file specification.
007D 120 :
007D 121 : mac$ab_tmsym adr The address of the assembler's temporary buffer
007D 122 : used to accumulate the qualifier name(s) and
007D 123 : module name(s) specified in an include list.
007D 124 :
007D 125 : mac$gq_lnkopt adr The address of the head of the linker options
007D 126 : records queue.
007D 127 :
007D 128 : Implicit Outputs:
007D 129 :
007D 130 : Linker option record(s) are placed into an ordered queue
007D 131 : (mac$gq_lnkopt) where the order is preserved FIFO. This is done
007D 132 : to remain compatible with the LINKER's normal processing of option
007D 133 : records (as if they were specified in a normal linker options file).
007D 134 : The list is subsequently written to the object file during pass 2,
007D 135 : following the object module header information.
007D 136 :
007D 137 : Special Case(s):
007D 138 :
007D 139 : Special processing is performed when the /INCLUDE qualifier is specified.
007D 140 : The object module names contained in the include list are constructed as
007D 141 : .ASCII strings following the filespec. The record is terminated by a
007D 142 : module name string with a zero length.
007D 143 :
007D 144 : If both the /LIBRARY and /INCLUDE qualifiers are specified for the
007D 145 : same library filespec, then the linker option record type is defaulted
007D 146 : to LNK$C_OLI and a special flag bit is set to indicate that the library
007D 147 : may be searched (LNK$V_LIBSRCH).
007D 148 :
007D 149 : Side Effects:
007D 150 :
007D 151 : Two possible side effects can occur. The first is a recoverable syntax
007D 152 : error where a diagnostic is issued to inform the user of the problem and
007D 153 : the assembly of the current input file continues. The second is an
007D 154 : insufficient virtual memory error in which a diagnostic is issued to the
007D 155 : user and the assembly of the current input file is aborted.
007D 156 :

```

```
007D 157 : Stack Usage:
007D 158 :
007D 159 :      31      16      8      0
007D 160 : +-----+-----+-----+
007D 161 : |      flags      ! lnktyp! rectyp!
007D 162 : +-----+-----+-----+
007D 163 : | module name include list flink!
007D 164 : +-----+-----+-----+
007D 165 : | module name include list blink!
007D 166 : +-----+-----+-----+
007D 167 : | #bytes req for inclst
007D 168 : +-----+-----+-----+
007D 169 : | link directive states
007D 170 : +-----+-----+-----+
007D 171 : --
007D 172 :
00000000 173 .psect mac$ro_code_p1, nowrt, gbl, long
0000 174
0000 175 link:: : Directive = klink
0000 176 : Save registers.
58 0100 8F BB 0000 176 pushr #^m<r8>
0000 177 movab g^mac$ab_tmpbuf,r8 : Address of temporary buffer.
0000 178 movab 8(r8),dsc$a_pointer(r8) : Address of buffer to accumulate file name.
5E 08 A8 9E 000B 178 movab -lnk_l_states(sp), sp : Allocate STACK LOCAL storage.
10 AE D4 0015 179 movab lnk_l_states(sp) : Initialize state flags.
0018 180
0018 181
0018 182 :
0018 183 : Perform initialization...then get the [next] file name.
0018 184
0018 185 10$: clrl (sp) : Reset linker record type and flags.
0018 186 clrl lnk_l_bytes (sp) : Reset the include list byte count.
04 AE 0C AE D4 001A 186 movab lnk_q_inclst (sp), lnk_q_inclst (sp) : Reset the queue list
08 AE 04 AE 9E 001D 187 movab lnk_q_inclst (sp), lnk_q_inclst+4 (sp) : head to empty condition.
6E 06 90 0027 188 movb #obj$c_ln,lnk$b_rectyp(sp) : Set the record type field.
58 DD 002A 189 pushl r8 : Push address of descriptor/buffer.
000000C3'EF 01 FB 002C 190 calls #1, get_file_name : Get the [next] file name (if any).
08 50 E8 0033 191 blbs r0, 20$ : We got one...go check for qualifier(s).
67 10 AE 00 E0 0036 192 bbs #0,lnk_l_states(sp),70$ : Have we processed at least one file?
0073 31 003B 193 brw 90$ : No, unterminated macro argument error.
003E 194
003E 195
003E 196 :
003E 197 : The file name has been accumulated, check for qualifiers.
003E 198
003E 199 20$: tstw (r8) : Null file specification?
00000000'GF 13 0040 200 beql 80$ : Yes, directive syntax error.
00000000'GF 16 0042 201 jsb g^mac$getchr : Skip over delimiter.
2F 5A 91 004E 202 jsb g^mac$skipsp : Skip spaces
11 13 0051 203 cmpb r10,#^A"/ : Was the character a slash?
2C 5A 91 0053 204 beql 25$ : Yes, process the qualifier(s).
1A 13 0056 205 cmpb r10,#^A", : No, was it a comma (valid syntax)?
2D 5A 91 0058 206 beql 30$ : Yes, use defaults.
15 13 005B 207 cmpb r10,#^A"- : Is the line continued?
0D 5A 91 005D 208 beql 30$ : Yes, finish current spec before continuing
10 13 0060 209 cmpb r10,#cr : Have we reached the end of the input line?
46 11 0062 210 beql 30$ : Yes, normal terminator.
5E DD 0064 211 brb 80$ : No, syntax error.
00000106'EF 01 FB 0066 212 25$: pushl sp : Push the address of the link vector.
213 calls #1, process_qual : Process the qualifier(s).
```

```
06 50 E8 006D 214 blbs r0 40$ ; Use the option record type scanned.
38 11 0070 215 brb 80$ ; Syntax error in qualifier(s).
0072 216
0072 217
0072 218 ; Build the linker option record.
0072 219
01 AE 03 90 0072 220 30$: movb #lnk$c_obj,lnk$b_lnktyp(sp) ; Default option record type.
5E DD 0076 221 40$: pushl sp ; Build a linker record using the
58 DD 0078 222 pushl r0 ; type and flags scanned above
00000299'EF 02 FB 007A 223 calls #2, build_lnk_rec ; (Note: state flags are affected).
0081 224
0081 225
0081 226 ; What's our next move...?
0081 227
2C 5A 91 0081 228 cmpb r10, #^A',' ; Is the current character a comma?
08 13 0084 229 beql 50$ ; Yes, go process the next list item.
2D 5A 91 0086 230 cmpb r10, #^A'-' ; Is the line continued?
12 12 0089 231 bneq 60$ ; No, check for eol.
5A 0D 90 008B 232 movb #cr, r10 ; Yes, continue processing the directive.
00000000'GF 16 008E 233 50$: jsb g^mac$getchr ; Skip current character.
00000000'GF 16 0094 234 jsb g^mac$skipsp ; Skip spaces, tabs, etc..
FF7B 31 009A 235 brw 10$ ; Continue
0D 5A 91 009D 236 60$: cmpb r10,#cr ; Have we reached the end of the line?
08 12 00A0 237 bneq 80$ ; No, report syntax error.
00A2 238
00A2 239
00A2 240 ; We're done, clean up and return to the parser.
00A2 241
5E 10 C0 00A2 242 70$: addl #lnk_l_states, sp ; Restore the stack.
0100 8F BA 00A5 243 popr #^m<785 ; Restore registers.
05 05 00A9 244 rsb ; Return to parser to continue.
00AA 245
00AA 246
00AA 247 ; Some type of syntax error has been encountered...
00AA 248
00AA 249 80$: $mac_err dirsyntax ; A directive syntax error has been
05 11 00AF 250 brb 100$ ; encountered, issue error and return.
00B1 251
00B1 252 90$: $mac_err untermarg ; A terminator has not been seen for
00B6 253 ; the file name.
00B6 254
5E 10 C0 00B6 255 100$: addl #lnk_l_states, sp ; Restore the stack.
0100 8F BA 00B9 256 popr #^m<785 ; Restore current token.
00000000'GF 17 00BD 257 jmp g^mac$errorpt ; Issue error message and return.
00C3 258
```

```
00C3 260 .sbttl get_file_name accumulate file name
00C3 261 :++
00C3 262 :
00C3 263 : Functional Description:
00C3 264 :
00C3 265 : This routine scans the input record for a delimited file name.
00C3 266 :
00C3 267 : Inputs:
00C3 268 :
00C3 269 : 4(AP) adr The address of a descriptor which points to a buffer
00C3 270 : to store the file name which we will scan.
00C3 271 :
00C3 272 : Outputs:
00C3 273 :
00C3 274 : 4(AP) adr The descriptor has been updated to reflect the size
00C3 275 : of the file name which has been accumulated.
00C3 276 :
00C3 277 : Routine Value:
00C3 278 :
00C3 279 : True A file name has been scanned.
00C3 280 : False No file name has been found (end of line or unterminated arg).
00C3 281 :
00C3 282 : Side Effects:
00C3 283 :
00C3 284 : If there is no file name available (eg. we hit the end of the line)
00C3 285 : the length field of the descriptor will be zero upon exit or if the
00C3 286 : argument is unterminated, the length will be non-zero but the routine
00C3 287 : value will be false.
00C3 288 :
00C3 289 :--
0044 00C3 290 .entry get_file_name ^m<r2,r6> ; Save registers upon entry.
51 04 AC DD 00C5 291 pushl r1 ; Preserve R1.
52 0000 C1 D0 00C7 292 movl 4(ap), r1 ; Get descriptor address.
00000000 GF 16 00CB 293 clrl (r1) ; Initialize length, class, and type fields.
0D 5A 91 00CD 294 movl dsc$a_pointer(r1), r2 ; Get the buffer address.
56 5A 90 00D2 295 clrl r0 ; Assume the worst...
6B 01 C8 00D4 296 jsb g^mac$skipsp ; Find the delimiter.
00000000 GF 16 00DA 297 cmpb r10,#cr ; Have we reached the end of the line?
56 5A 91 00DD 298 beql 30$ ; Yes, return FALSE to the caller.
0D 5A 90 00DF 299 movb r10,r6 ; No, copy the delimiter and pass semi
6B 01 C8 00E2 300 bisl2 #flg$m_allchr,(r11) ; colons (to allow a version number).
00000000 GF 16 00E5 301
00000000 GF 16 00E5 302 10$: jsb g^mac$getchr ; Get the next character of the filename.
56 5A 91 00EB 303 cmpb r10,r6 ; Is it the delimiter (end of filename)?
0D 5A 91 00EE 304 beql 20$ ; Yes, we're done here, return.
0D 5A 91 00F0 305 cmpb r10,#cr ; No, is it the end of the line?
82 5A 90 00F3 306 beql 30$ ; Yes, upon return issue unterminated argume
61 B6 00F5 307 movb r10,(r2)+ ; No, store the character.
E9 11 00F8 308 incw (r1) ; Keep track of file name length.
00FC 309 brb 10$ ; Gather the rest of the file name.
50 01 D0 00FC 310
51 8ED0 00FF 311 20$: movl #1, r0 ; Success...
6B 01 CA 0102 312 30$: popl r1 ; Restore R1.
0105 313 bicl2 #flg$m_allchr,(r11) ; Don't pass anymore semi-colons...
0106 314 ret
0106 315
```

```
0106 317 .sbtll process_qual process link directive qualifiers
0106 318 :++
0106 319 :
0106 320 : Functional Description:
0106 321 :
0106 322 : This routine processes the .LINK directive qualifiers.
0106 323 :
0106 324 : Inputs:
0106 325 :
0106 326 : 4(AP) adr Address of a linker record vector.
0106 327 :
0106 328 : Outputs:
0106 329 :
0106 330 : 4(AP) adr The linker record information is set in the vector.
0106 331 :
0106 332 : Routine Value:
0106 333 :
0106 334 : True Qualifiers have been processed without a problem.
0106 335 : False There was a syntax error in either the qualifier name or in
0106 336 : the item list associated with the qualifier.
0106 337 :
0106 338 :--
0106 339 .entry process_qual ^m<r5,r6,r7,r8> ; Save registers.
5E F8 AE 9E 0108 340 movab -8(sp), sp ; Get STACK LOCAL storage.
0106 341 clrl (sp) ; Initialize done bit.
57 04 AC D0 010E 342 movl 4(ap), r7 ; Get base adr of link info vector.
58 04 A7 9E 0112 343 movab lnk_q_inclst (r7), r8 ; Base address of file name list hea
0106 344 :
0106 345 10$: cmpb r10, #^A/,/ ; Did we stop on a comma?
0106 346 beql 30$ ; Yes, we're done with this file spe
0106 347 cmpb r10, #cr ; No, have we reached eol?
0106 348 beql 30$ ; Yes, we're done, return.
0106 349 cmpb r10, #^A'/' ; Is the current character slash?
0106 350 beql 40$ ; Yes, scan qualifier name.
0106 351 20$: brw 110$ ; No, syntax error.
0106 352 30$: brw 100$ ; Done, return success.
0106 353 :
0106 354 40$: jsb g^mac$getchr ; Yes, skip over it...
0106 355 jsb g^mac$symscnup ; Get the qualifier name.
0106 356 blbc r0, 20$ ; None found, error.
55 00000000'GF 16 0137 356 blbc r0, 20$ ; None found, error.
0106 357 movab lnk_qualifiers, r5 ; Use linker qualifier name table.
0106 358 jsb g^mac$src_list ; Look up linker option qualifier.
0106 359 blbc r0, 20$ ; Not found, error.
0106 360 jsb g^mac$skipsp ; Position character pointer as need
0106 361 :
0106 362 : Dispatch to appropriate processing routine.
0106 363 :
0106 364 :
04 00 05 A1 CF 0150 365 case1 sym$l_val(r1), #0, #lnk$c_maxrectyp
0106 366 50$: .word 60%-50$ ; lnk$c_olb - /LIBRARY
0106 367 .word 70%-50$ ; lnk$c_shr - (unsupported)
0106 368 .word 80%-50$ ; lnk$c_oli - /INCLUDE=
0106 369 .word 90%-50$ ; lnk$c_obj - /SELECTIVE_SEARCH
0106 370 .word 70%-50$ ; lnk$c_sha - /SHAREABLE
0106 371 brb 95$ ; Default, OBJ or STB
0106 372
```

```
0161 374 :  
0161 375 : /LIBRARY Normal object library  
0161 376 :  
01 A7 95 0161 377 60$: tstb lnk$b_lnktyp (r7) : Check for conflicting qualifiers.  
06 13 0164 378 beql 63$ : None specified.  
02 01 A7 91 0166 379 cmpb lnk$b_lnktyp (r7), #lnk$sc_oli : If /INCLUDE was specified, no conf  
65 12 016A 380 bneq 110$ : but anything else will conflict.  
01 A7 00 90 016C 381 63$: movb #lnk$sc_olb, lnk$b_lnktyp (r7) : lnk$sc_olb - Normal object library.  
00 6E 00 E2 0170 382 bbss #0, (sp), +1 : Indicate /LIBRARY has been specifi  
08 6E 01 E1 0174 383 bbc #1, (sp), 65$ : If /INCLUDE has been specified, th  
02 A7 02 88 0178 384 bisb #lnk$sm_libsrch, lnk$w_flags (r7) : the library should be searched and  
01 A7 02 90 017C 385 movb #lnk$sc_oli, lnk$b_lnktyp (r7) : type precedence goes to LNK$C_OLI.  
FF93 31 0180 386 65$: brw 10$ : Get the next entity.  
0183 387 :  
0183 388 : /SHAREABLE Shareable Image  
0183 389 :  
01 A7 95 0183 391 70$: tstb lnk$b_lnktyp (r7) : Check for conflicting qualifiers.  
49 12 0186 392 bneq 110$ : We have a conflict.  
01 A7 04 90 0188 393 movb #lnk$sc_sha, lnk$b_lnktyp (r7) : lnk$sc_sha - Shareable Image  
FF87 31 018C 394 brw 10$ : Get the next entity.  
018F 395 :  
018F 396 : /INCLUDE Object Library with Include List  
018F 397 :  
01 A7 95 018F 399 80$: tstb lnk$b_lnktyp (r7) : Check for conflicting qualifiers.  
06 13 0192 400 beql 82$ : None specified.  
00 01 A7 91 0194 401 cmpb lnk$b_lnktyp (r7), #lnk$sc_olb : If /LIBRARY was specified, no conf  
37 12 0198 402 bneq 110$ : but anything else will conflict.  
01 A7 02 90 019A 403 82$: movb #lnk$sc_oli, lnk$b_lnktyp (r7) : lnk$sc_oli - Object Library with an  
00 6E 01 E2 019E 404 bbss #1, (sp), +1 : Indicate /INCLUDE has been specifi  
04 6E 00 E1 01A2 405 bbc #0, (sp), 83$ : If /LIBRARY has been specified, th  
02 A7 02 88 01A6 406 bisb #lnk$sm_libsrch, lnk$w_flags (r7) : the library should be searched.  
3D 5A 91 01AA 407 83$: cmpb r10, #A/= : The next character should be an '='  
000001D4'EF 00 FB 01AF 409 calls #0, get_incl_list : If not, its a syntax error.  
18 50 E9 01B6 410 blbc r0, 110$ : Get the module name(s) in the incl  
FF5A 31 01B9 411 85$: brw 10$ : Issue syntax error.  
01BC 412 : Get the next entity.  
01BC 413 :  
01BC 414 : /SELECTIVE_SEARCH Selective search of OLB or STB  
01BC 415 :  
01 A7 95 01BC 416 90$: tstb lnk$b_lnktyp (r7) : Check for conflicting qualifiers.  
10 12 01BF 417 bneq 110$ : We have a conflict.  
02 A7 01 88 01C1 418 bisb #lnk$sm_selser, lnk$w_flags (r7) : lnk$sm_selser - Selective search  
01 A7 03 90 01C5 419 95$: movb #lnk$sc_obj, lnk$b_lnktyp (r7) : lnk$sc_obj - Object Module  
FF4A 31 01C9 420 brw 10$ : Get the next entity.  
01CC 421 :  
01CC 422 :  
01CC 423 : All done select status and return.  
01CC 424 :  
50 01 D0 01CC 425 100$: movl #1, r0 : Success.  
02 11 01CF 426 brb 120$ : Now return.  
50 D4 01D1 427 110$: clrl r0 : Error.  
04 01D3 428 120$: ret : Restore registers and return.  
01D4 429
```

```
01D4 431 .sbtll get_incl_list Get the module(s) in the list
01D4 432 :++
01D4 433 :
01D4 434 Functional Description:
01D4 435 :
01D4 436 This routine scans the include list and produces a linked list
01D4 437 containing the counted ascii strings of the module name(s) entered
01D4 438 in the include list.
01D4 439 :
01D4 440 Implicit Inputs:
01D4 441 :
01D4 442 r7 The address of the linker record.
01D4 443 r8 The address of the module name include list head.
01D4 444 :
01D4 445 mac$ab_tmsym The address of a symbol just scanned.
01D4 446 :
01D4 447 Outputs:
01D4 448 :
01D4 449 Module names obtained from the include list are added to the linked
01D4 450 list.
01D4 451 :
01D4 452 :--
01D4 453 .entry get_incl_list ^m<> ; Get the module names to include.
5E FC AE 0000 01D6 454 movab -4(sp), sp ; Get LOCAL STORAGE.
6E D4 01DA 455 clrl (sp) ; Initialize local storage.
01DC 456 :
00000000'GF 16 01DC 457 10$: jsb g^mac$getchr ; Get the next character.
00000000'GF 16 01E2 458 jsb g^mac$skipsp ; Skip spaces, tabs, etc..
UD 5A 91 01E8 459 cmpb r10, #cr ; End of line?
54 13 01EB 460 beql 80$ ; Yes, syntax error.
01ED 461 :
28 5A 91 01ED 462 20$: cmpb r10, #^A/(/ ; Do we have a list of names?
25 13 01F0 463 beql 30$ ; Yes, remove open paren and indicat
2C 5A 91 01F2 464 cmpb r10, #^A/,/ ; Check for module name delimiter.
26 13 01F5 465 beql 40$ ; Remove the comma, and validate str
29 5A 91 01F7 466 cmpb r10, #^A/)/ ; Do we have a close paren?
2B 13 01FA 467 beql 50$ ; Yes, end of the list?
OD 5A 91 01FC 468 cmpb r10, #cr ; End of line?
37 13 01FF 469 beql 60$ ; We're done, select return status.
00 6E 00 E4 0201 470 bbsc #0, (sp), .+1 ; Reset comma seen flag.
00000000'GF 16 0205 471 jsb g^mac$symscnup ; Get the module name.
33 50 E9 020B 472 blbc r0, 80$ ; No file name, error.
0000024B'EF 00 FB 020E 473 calls #0, insert_module ; Insert this module name into the l
D6 11 0215 474 brb 20$ ; Get the next module (if any).
0217 475 :
26 6E 01 E2 0217 476 30$: bbss #1, (sp), 80$ ; Check for syntax error -- 2 or mor
BF 11 021B 477 brb 10$ ; Indicate a list and continue the m
021D 478 :
20 6E 01 E1 021D 479 40$: bbc #1, (sp), 80$ ; Comma seperated list not allowed o
1C 6E 00 E2 0221 480 bbss #0, (sp), 80$ ; To many commas?
B5 11 0225 481 brb 10$ ; Remove comma, step to next module
0227 482 :
16 6E 01 E1 0227 483 50$: bbc #1, (sp), 80$ ; Should we have a close paren?
00000000'GF 16 022B 484 jsb g^mac$getchr ; Yes, skip it for correct grammatic
68 58 D1 0231 485 cmpl r8, (r8) ; Have we got at least one module?
OB 13 0234 486 beql 80$ ; No, and we don't accept null lists
04 11 0236 487 brb 70$ ; Everything looks ok, return succes
```

05	6E	01	E0	0238	488						
				0238	489	60\$:	bbs	#1, (sp), 80\$; Should we have a close paren? (pre	
				023C	490						
	50	01	D0	023C	491	70\$:	movl	#1, r0		; Parse successful...	
		09	11	023F	492		brb	90\$; return success.	
				0241	493						
		50	D4	0241	494	80\$:	clrl	r0		; Parse failed.	
	68	58	D0	0243	495		movl	r8, (r8)		; Fake an empty queue...	
04	A8	58	D0	0246	496		movl	r8, 4(r8)		; reset the list head.	
			04	024A	497	90\$:	ret			; Return	
				024B	498						

```
024B 500 .sbtll insert_module Insert module name into list
024B 501 :++
024B 502 :
024B 503 : Functional Descrtipion:
024B 504 :
024B 505 : This routine will allocate a block of memory to store the
024B 506 : module name (counted asci string) and link it into the list
024B 507 : of other modules in the include list.
024B 508 :
024B 509 : Implicit Inputs:
024B 510 :
024B 511 : r7      adr      Address of the linker options record control block.
024B 512 : r8      adr      Address of the head of the include list.
024B 513 :
024B 514 : mac$ab_tmptsym  The buffer containing the counted asci string.
024B 515 :
024B 516 : Outputs:
024B 517 :
024B 518 : The module has been added to the list.
024B 519 :
024B 520 : Side Effects:
024B 521 :
024B 522 : If an error occurs while attempting to allocate dyanmic memory
024B 523 : we'll exit in the normal tradition (abort this assembly).
024B 524 :
024B 525 :--
027C 024B 526 .entry insert_module ^m<r2,r3,r4,r5,r6,r9>
56 00000000'GF 51 DD 024D 527 pushl r1 ; Preserve R1.
59 66 9A 024F 528 movab g^mac$ab_tmptsym, r6 ; Get the beginning address of the buffer.
59 59 D6 0256 529 movzbl (r6), r9 ; Get the size of the string.
5E 08 C2 0259 530 incl r9 ; Bump the string count to include count byt
04 AE 04 AE D4 025B 531 subl2 #8, sp ; Get STACK LOCAL storage.
04 AE 04 AE 9F 025E 532 clrl 4(sp) ; Initialize the return address scalar.
04 AE 59 08 C1 0261 533 pushab 4(sp) ; Push address of the return address scalar.
04 AE 04 AE 9F 0264 534 addl3 #8, r9, 4(sp) ; Compute the number of bytes to allocate.
00000000'GF 02 FB 0269 535 pushab 4(sp) ; Push the address of the number of bytes.
16 50 E9 026C 536 calls #2, g^lib$get_vm ; Allocate the module name block.
53 04 AE D0 0273 537 blbc r0, ins_vir_mem ; Insufficient Vitrual Memory error.
08 A3 66 59 28 0276 538 movl 4(sp), r3 ; Get the beginning address of the module na
04 B8 04 BE 0E 027A 539 movc3 r9, (r6), 8(r3) ; Copy the string.
0C A7 59 C0 027F 540 insque @4(sp), @4(r8) ; Insert this block at the tail of the list.
51 8ED0 0284 541 addl2 r9, lnk_l_bytes (r7) ; Add this string's byte count to the sum.
04 0288 542 popl r1 ; Restore R1.
028B 543 ret
028C 544
028C 545 ;
028C 546 ; Insufficent Virtual Memory, report error and abort this assembly.
028C 547 ;
028C 548 ins_vir_mem:
00000000'GF 00 FB 028C 549 calls #0, g^mac$err_nomem_0 ; Report error.
00000000'GF 17 0293 550 jmp g^mac$last_chance ; Abort this assembly.
0299 551
```

```

0299 553      .sbttl  build_lnk_rec  Build a linker options record
0299 554      ;++
0299 555
0299 556      Functional Description:
0299 557
0299 558      This routine builds a linker options record.  First we obtain
0299 559      a linker options record block (by a call to LIB$GET_VM), next
0299 560      fill in the information and link it into the queue of linker
0299 561      options records and set the 'at least one file processed' bit
0299 562      in the state flags.  Special handling of the object library with
0299 563      an include list is performed by copying the strings from the
0299 564      include module list and deallocating the module name blocks.
0299 565
0299 566      31          16          8          0
0299 567      +-----+-----+-----+-----+
0299 568      |                               | flink |
0299 569      +-----+-----+-----+-----+
0299 570      |                               | blink |
0299 571      +-----+-----+-----+-----+
0299 572      | !size of this lnk opt rec block! |
0299 573      +-----+-----+-----+-----+
0299 574      | flags          | lnktyp! rectyp! |
0299 575      +-----+-----+-----+-----+
0299 576      |                               | namlng |
0299 577      |               +-----+-----+ |
0299 578      |               | file specification |
0299 579      +-----+-----+-----+-----+
0299 580      |                               | count |
0299 581      | optional module |-----+-----+ |
0299 582      | name from /INCLUDE=[()] list |
0299 583      +-----+-----+-----+-----+
0299 584      |                               |
0299 585      |                               |
0299 586      +-----+-----+-----+-----+
0299 587      |                               | count |
0299 588      | optional module |-----+-----+ |
0299 589      | name from /INCLUDE=[()] list |
0299 590      +-----+-----+-----+-----+
0299 591      |                               | 0 | Terminator byte (module name length 0)
0299 592      +-----+-----+-----+-----+
0299 593
0299 594      Inputs:
0299 595
0299 596      4(AP)      adr      Address of the filespec descriptor.
0299 597      8(AP)      adr      Address of the linker record control block.
0299 598
0299 599      Implicit Inputs:
0299 600
0299 601      mac$gq_lnkopt  adr      The address of the linker options record queue
0299 602      list head.
0299 603
0299 604      Side Effects:
0299 605
0299 606      If an error occurs while attempting to allocate dyanmic memory
0299 607      we'll exit in the normal tradition (abort this assembly).
0299 608      --

```

```
01FC 0299 610 .entry build_lnk_rec ^m<r2,r3,r4,r5,r6,r7,r8>
52 04 AC D0 029B 611 movl 4(ap), r2 ; Get the address of the filespec.
57 08 AC D0 029F 612 movl 8(ap), r7 ; Get link control block address.
5E 08 C2 02A3 613 subl2 #8, sp ; Allocate STACK LOCAL storage.
04 AE 04 AE D4 02A6 614 clrl 4(sp) ; Initialize the return address scalar.
04 AE 04 AE 9F 02A9 615 pushab 4(sp) ; Push the address of the return address sca
04 AE 0C A7 04 AE C2 3C 02AC 616 movzwl dsc$w_length(r2), 4(sp) ; Get the length.
04 AE 04 AE 13 C0 02B2 617 addl3 4(sp), lnk_l_bytes(r7), 4(sp) ; Compute the #bytes req for this record
04 AE 04 AE 9F 02B9 618 addl2 #lnk_blk_si2, 4(sp) ; Include the fixed area size in the count.
00000000'GF 02 FB 02C0 619 pushab 4(sp) ; Address of the number of byte req.
53 04 AE D0 02C7 620 calls #2, g^lib$get_vm ; Allocate memory for this linker options re
58 53 D0 02CA 621 blbc r0, ins_vir_mem ; Insufficient Virtual Memory error.
53 08 C0 02D1 622 movl 4(sp), r3 ; Get the beginning address of the linker op
83 6E D0 02D4 623 movl r3, r8 ; Preserve the block address for later use.
83 67 D0 02D7 624 addl2 #8, r3 ; Advance pointer to first data field.
83 62 B0 02DA 625 movl (sp), (r3)+ ; Copy the number of bytes in this block.
63 0000'D2 62 28 02DD 626 movl (r7), (r3)+ ; Copy the record type and flags word to the
00000000'GF 02 FB 0303 627 movw (r2), (r3)+ ; Copy the file spec length.
56 04 B7 0F 02E3 628 movc3 (r2), @dsc$a_pointer(r2), (r3) ; Copy the file spec.
26 1D 02E7 629 remque @lnk_q_inclst(r7), r6 ; Remove the next module name.
04 AE 56 D0 02E9 630 bvs 20$ ; Is the queue empty?
04 AE 04 AE 9F 02ED 631 movl r6, 4(sp) ; No, get the address of the module name
04 AE 08 A6 9A 02F0 632 pushab 4(sp) ; block to release and pass it by reference.
04 AE 01 80 02F5 633 movzbl 8(r6), 4(sp) ; Get the string length and include
63 08 A6 04 AE 28 02F9 634 addb2 #1, 4(sp) ; the count byte in the string size.
04 AE 08 C0 02FF 635 movc3 4(sp), 8(r6), (r3) ; Copy the module size/name to the record.
04 AE 9F 0303 636 addl2 #8, 4(sp) ; The linkage is included in the mnb size.
00000000'GF 02 FB 0306 637 pushab 4(sp) ; Pass the block size by reference too.
D4 11 030D 638 calls #2, g^lib$free_vm ; Release the module name block.
63 94 030F 639 brb 10$ ; Get the next module.
56 00000000'GF 9E 0311 640 clrb (r3) ; Mark include list terminator.
04 B6 68 0E 0318 641 movab g^mac$gq_lnkopt, r6 ; Get the address of the linker options queu
00 10 A7 00 E2 031C 642 insque (r8), @4(r6) ; Insert this record into the linker options
04 0321 643 bbss #0, lnk_l_states(r7), ..+1 ; Set flag indicating at least 1 file proces
0322 644 ret
647
```

```
0322 649 .sbtll mac$wrt_lnkopt Write the linker options records to object
0322 650 :++
0322 651
0322 652 Functional Description:
0322 653
0322 654 This routine removes the linker options records from the queue
0322 655 MAC$GQ_LNKOPT and writes them to the object module (following the GSD).
0322 656
0322 657 Implicit Inputs:
0322 658
0322 659 R10 adr Contains the address of the object code buffer.
0322 660 mac$gq_lnkopt adr The address of the linker option record queue.
0322 661
0322 662 Side Effects:
0322 663
0322 664 All linker option record(s) have been written to the object file
0322 665 and the current object record buffer type will be set to OBJ$C_TIR
0322 666 upon exit.
0322 667 :--
0322 668
0322 669 .entry mac$wrt_lnkopt ^m<r6> ; Save register(s).
0324 670 subl2 #8, sp ; Allocate STACK LOCAL storage.
0327 671 10$: remque @mac$gq_lnkopt, r6 ; Get a linker option record.
032E 672 bvs 20$ ; Is the queue empty?
0330 673 decl r10 ; No, set the buffer pointer to origin.
0332 674 subl3 #12, 8(r6), r1 ; Compute the size of this record.
0337 675 movc3 r1, 12(r6), (r10) ; Copy the record to the object code buffer.
033C 676 movl r3, r10 ; Update the object code pointer.
033F 677 bsbw mac$wrtobj ; Write the object record.
0342 678 movl r6, 4(sp) ; Release dynamic memory...
0346 679 pushab 4(sp) ; Pass the block's address by reference.
0349 680 movl 8(r6), 4(sp) ; The linker option record's block size
034E 681 pushab 4(sp) ; is also passed by reference.
0351 682 calls #2, g^lib$free_vm ; Release this block.
0358 683 brb 10$ ; Continue until the queue is empty.
035A 684
035A 685 20$: movb #obj$c_tir, (r10) ; All done, correct the object record
035D 686 ret ; type to assume TIR.
035E 687
035E 688 .end ; of MODULE mac$link
```

56 00000000'FF 0F 0327 671 10\$: remque @mac\$gq_lnkopt, r6 ; Get a linker option record.

51 08 A6 0C C3 0332 674 subl3 #12, 8(r6), r1 ; Compute the size of this record.

6A 0C A6 51 28 0337 675 movc3 r1, 12(r6), (r10) ; Copy the record to the object code buffer.

5A 53 D0 033C 676 movl r3, r10 ; Update the object code pointer.

04 AE 56 D0 033F 677 bsbw mac\$wrtobj ; Write the object record.

04 AE 08 A6 D0 0342 678 movl r6, 4(sp) ; Release dynamic memory...

04 AE 04 AE 9F 0346 679 pushab 4(sp) ; Pass the block's address by reference.

00000000'GF 02 FB 0349 680 movl 8(r6), 4(sp) ; The linker option record's block size

CD 11 034E 681 pushab 4(sp) ; is also passed by reference.

6A 02 90 0351 682 calls #2, g^lib\$free_vm ; Release this block.

04 0358 683 brb 10\$; Continue until the queue is empty.

035A 684

035A 685 20\$: movb #obj\$c_tir, (r10) ; All done, correct the object record

035D 686 ret ; type to assume TIR.

035E 687

035E 688 .end ; of MODULE mac\$link

MACSLINK
Symbol table

link directive processor

H 4

16-SEP-1984 02:06:27
5-SEP-1984 01:48:43VAX/VMS Macro V04-00
[MACRO.SRC]LINK.MAR;1Page 15
(11)

```
BUILD_LNK_REC = 00000299 RG 05
CR = 0000000D
DSCSA_POINTER ***** X 05
DSCSW_LENGTH ***** X 05
EOMSC_ABORT = 00000003
EOMSC_ERROR = 00000002
EOMSC_SUCCESS = 00000000
EOMSC_WARNING = 00000001
FLGSM_ALLCHR = 00000001
FLGSM_BOL = 00000002
FLGSM_CHKLPND = 00100000
FLGSM_COMPEXPR = 00000004
FLGSM_CONT = 00000008
FLGSM_CRF = 40000000
FLGSM_CRSEEN = 00000001
FLGSM_DATRPT = 00000010
FLGSM_DBGOUT = 00004000
FLGSM_DLIMSTR = 00008000
FLGSM_ENDMCH = 00000020
FLGSM_EVALEXPR = 00000040
FLGSM_EXPOPT = 00000080
FLGSM_EXTERR = 00010000
FLGSM_EXTWRN = 00020000
FLGSM_FIRSTLN = 00000200
FLGSM_IFSTAT = 00800000
FLGSM_IIF = 00400000
FLGSM_INSERT = 00000100
FLGSM_IRPC = 20000000
FLGSM_LEXOP = 00000002
FLGSM_LSTXST = 00000200
FLGSM_MAC2COL = 00000800
FLGSM_MACL = 00000800
FLGSM_MACLTB = 08000000
FLGSM_MACTXT = 00010000
FLGSM_MEBLST = 00001000
FLGSM_MOREARG = 00002000
FLGSM_MOREINP = 00000008
FLGSM_NEWPND = 00000400
FLGSM_NOREF = 01000000
FLGSM_NTTYPEPC = 00000020
FLGSM_NULCHR = 00040000
FLGSM_OBJXST = 00200000
FLGSM_OPNDCHK = 00000100
FLGSM_OPRND = 00002000
FLGSM_OPTVFLIDX = 00001000
FLGSM_ORDLST = 00020000
FLGSM_P2 = 00004000
FLGSM_RPTIRP = 10000000
FLGSM_SEQFIL = 02000000
FLGSM_SKAN = 00008000
FLGSM_SPECOP = 00000004
FLGSM_SPLALL = 04000000
FLGSM_STOIMF = 00040000
FLGSM_SYM2COL = 00000400
FLGSM_TOCLFG = 00080000
FLGSM_UPAFLG = 00000010
FLGSM_UPDFIL = 00000080
```

```
FLGSM_UPMARG = 00000040
FLGSM_XCRF = 80000000
FLGSV_ALLCHR = 00000000
FLGSV_BOL = 00000001
FLGSV_CHKLPND = 00000014
FLGSV_COMPEXPR = 00000002
FLGSV_CONT = 00000003
FLGSV_CRF = 0000001E
FLGSV_CRSEEN = 00000020
FLGSV_DATRPT = 00000004
FLGSV_DBGOUT = 0000002E
FLGSV_DLIMSTR = 0000002F
FLGSV_ENDMCH = 00000005
FLGSV_EVALEXPR = 00000006
FLGSV_EXPOPT = 00000007
FLGSV_EXTERR = 00000030
FLGSV_EXTWRN = 00000031
FLGSV_FIRSTLN = 00000029
FLGSV_IFSTAT = 00000017
FLGSV_IIF = 00000016
FLGSV_INSERT = 00000008
FLGSV_IRPC = 0000001D
FLGSV_LEXOP = 00000021
FLGSV_LSTXST = 00000009
FLGSV_MAC2COL = 0000002B
FLGSV_MACL = 0000000B
FLGSV_MACLTB = 0000001B
FLGSV_MACTXT = 00000010
FLGSV_MEBLST = 0000000C
FLGSV_MOREARG = 0000002D
FLGSV_MOREINP = 00000023
FLGSV_NEWPND = 0000000A
FLGSV_NOREF = 00000018
FLGSV_NTTYPEPC = 00000025
FLGSV_NULCHR = 00000032
FLGSV_OBJXST = 00000015
FLGSV_OPNDCHK = 00000028
FLGSV_OPRND = 0000000D
FLGSV_OPTVFLIDX = 0000002C
FLGSV_ORDLST = 00000011
FLGSV_P2 = 0000000E
FLGSV_RPTIRP = 0000001C
FLGSV_SEQFIL = 00000019
FLGSV_SKAN = 0000000F
FLGSV_SPECOP = 00000022
FLGSV_SPLALL = 0000001A
FLGSV_STOIMF = 00000012
FLGSV_SYM2COL = 0000002A
FLGSV_TOCLFG = 00000013
FLGSV_UPAFLG = 00000024
FLGSV_UPDFIL = 00000027
FLGSV_UPMARG = 00000026
FLGSV_XCRF = 0000001F
GET_FILE_NAME 000000C3 RG 05
GET_INCL_LIST 000001D4 RG 05
INSERT_MODULE 0000024B RG 05
INSYMC = 00000007
```

```
INSYMP = 00000074 R 04
INSYTM = 00000074 R 04
INS_VIR_MEM 0000028C R 05
LIB$FREE_VM ***** X 05
LIB$GET_VM ***** X 05
LINK 00000000 RG 05
LNK$B_LNK_TYP = 00000001
LNK$B_RECTYP = 00000000
LNK$C_MAXRECTYP = 00000004
LNK$C_OBJ = 00000003
LNK$C_OLB = 00000000
LNK$C_OLI = 00000002
LNK$C_SHA = 00000004
LNK$M_LIBSRCH = 00000002
LNK$M_SELSE = 00000001
LNK$W_FLAGS = 00000002
LNK_BCK_SIZ = 00000013
LNK_L_BYTES = 0000000C
LNK_L_STATES = 00000010
LNK_QUALIFIERS 00000074 RG 04
LNK_Q_INCLST = 00000004
MAC$AB_TMPBUF ***** X 05
MAC$AB_TPMSYM ***** X 05
MAC$ERRORTPT ***** X 05
MAC$ERR_NOMEM_0 ***** X 05
MAC$GETCHR ***** X 05
MAC$GQ_LNKOPT 00000000 RG 03
MAC$LAST_CHANCE ***** X 05
MAC$SKIPSP ***** X 05
MAC$SRC_LIST ***** X 05
MAC$SYM$CNUPI ***** X 05
MAC$WRTOBJ ***** X 05
MAC$WRT_LNKOPT 00000322 RG 05
MAC$DIRSYNX = 007D906A
MAC$UNTERMARG = 007D922A
OBJ$C_EOM_ABORT = 00000003
OBJ$C_EOM_ERR = 00000002
OBJ$C_EOM_OK = 00000000
OBJ$C_EOM_WRN = 00000001
OBJ$C_LNK = 00000006
OBJ$C_TIR = 00000002
OPF$M_LASTOPR = 00002000
OPF$M_OPTEXP = 00001000
OPF$V_LASTOPR = 0000000D
OPF$V_OPTEXP = 0000000C
PROCESS_QUAL 00000106 RG 05
PSC$B_NAME 00000004
PSC$B_SEG 0000000C
PSC$B_UNUSED 0000000B
PSC$K_BLK$IZ 00000013
PSC$K_NO_OPTNS = 0000000A
PSC$L_CURLOC 0000000F
PSC$L_LINK 00000000
PSC$L_MAXLGTH 00000005
PSC$M_ABS = FFFFFFFF7
PSC$M_ALIGNFLG = 00004000
PSC$M_ALLOPTNS = 000003FF
```

MAC\$LINK
Symbol table

link directive processor

I 4

16-SEP-1984 02:06:27 VAX/VMS Macro V04-00
5-SEP-1984 01:48:43 [MACRO.SRC]LINK.MAR;1

Page 16
(11)

PSC\$M_BYTE = 00004000
PSC\$M_CON = FFFFFFFFB
PSC\$M_DEFAULT = 000001C8
PSC\$M_EXE = 000000C0
PSC\$M_GBL = 00000010
PSC\$M_LCL = FFFFFFFEF
PSC\$M_LIB = 00000002
PSC\$M_LONG = 00004800
PSC\$M_NOEXE = FFFFFFFBF
PSC\$M_NOPIC = FFFFFFFFE
PSC\$M_NORD = FFFFFFF7F
PSC\$M_NOSHR = FFFFFFFDF
PSC\$M_NOVEC = FFFFFFFDF
PSC\$M_NOWRT = FFFFFFFEF
PSC\$M_OVR = 00000004
PSC\$M_PAGE = 00006400
PSC\$M_PIC = 00000001
PSC\$M_QUAD = 00004C00
PSC\$M_RD = 00000080
PSC\$M_REL = 00000008
PSC\$M_SHR = 00000020
PSC\$M_USR = FFFFFFFFD
PSC\$M_VEC = 00000200
PSC\$M_WORD = 00004400
PSC\$M_WRT = 00000180
PSC\$S_ALIGNMENT = 00000004
PSC\$V_ALIGNFLG = 0000000E
PSC\$V_ALIGNMENT = 0000000A
PSC\$V_EXE = 00000006
PSC\$V_GBL = 00000004
PSC\$V_LIB = 00000001
PSC\$V_OVR = 00000002
PSC\$V_PIC = 00000000
PSC\$V_RD = 00000007
PSC\$V_REL = 00000003
PSC\$V_SHR = 00000005
PSC\$V_VEC = 00000009
PSC\$V_WRT = 00000008
PSC\$W_FLAG = 00000009
PSC\$W_OPTIONS = 0000000D
SYMSB_NAME = 00000004
SYMSB_SEG = 0000000C
SYMSB_TOKEN = 0000000B
SYMSF_DEF = 00000002
SYMSF_REL = 00000008
SYMSF_UNI = 00000004
SYMSF_VALIDATE = 00000010
SYMSF_WEAK = 00000001
SYMSK_BLKSIZE = 0000000D
SYMSK_MAXLEN = 0000001F
SYMSL_LINK = 00000000
SYMSL_VAL = 00000005
SYMSM_ABS = 00000010
SYMSM_ASN = 00000100
SYMSM_CRFO = 00002000
SYMSM_DEBUG = 00000020
SYMSM_DEF = 00000001

SYMSM_DELMAC = 00000200
SYMSM_EPT = 00000200
SYMSM_EXTRN = 00000008
SYMSM_GLOBL = 00000004
SYMSM_LOCAL = 00000040
SYMSM_ODBG = 00000400
SYMSM_REF = 00000080
SYMSM_RELPSECT = 00000800
SYMSM_SUPR = 00004000
SYMSM_WEAK = 00000002
SYMSM_XCRF = 00001000
SYMSV_ABS = 00000004
SYMSV_ASN = 00000008
SYMSV_CRFO = 0000000D
SYMSV_DEBUG = 00000005
SYMSV_DEF = 00000000
SYMSV_DELMAC = 00000009
SYMSV_EPT = 00000009
SYMSV_EXTRN = 00000003
SYMSV_GLOBL = 00000002
SYMSV_LOCAL = 00000006
SYMSV_ODBG = 0000000A
SYMSV_REF = 00000007
SYMSV_RELPSECT = 0000000B
SYMSV_SUPR = 0000000E
SYMSV_WEAK = 00000001
SYMSV_XCRF = 0000000C
SYMSW_FLAG = 00000009
TIRSC_STO_L = 00000016
TIRSC_STO_LW = 00000016
X1 = 00000400
X2 = 0000000F

MAC
V04

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ABSS	00000013 (19.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MACSRW_DATA	00000008 (8.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
MAC\$RO_DATA	0000007D (125.)	04 (4.)	NOPIC USR CON REL GBL NOSHR NOEXE RD NOWRT NOVEC LONG
MAC\$RO_CODE_P1	0000035E (862.)	05 (5.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	32	00:00:00.06	00:00:00.46
Command processing	108	00:00:00.38	00:00:03.67
Pass 1	340	00:00:07.38	00:00:31.50
Symbol table sort	0	00:00:00.82	00:00:02.10
Pass 2	137	00:00:01.81	00:00:03.82
Symbol table output	22	00:00:00.12	00:00:00.12
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	644	00:00:10.60	00:00:41.70

Macro library name	Macros defined
-----	-----
-\$255\$DUA28:[MACRO.OBJ]MACRO.MLB;1	6
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	10

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:LINK/OBJ=OBJ\$:LINK MSRC\$:LINK/UPDATE=(ENH\$:LINK)+LIB\$:MACRO/LIB

MAC	Sym
ADD	
AIN	
AIN	
AIN	
ALI	
ALI	
ALI	
ASC	
ASC	
ASC	
ASS	
ASS	
ASS	
AUT	
AUT	
AUT	
BDE	
BDE	
BLK	
BLK	
BLK	
BLK	
BST	
BST	
BYT	
CHA	
CHR	
CHR	
CRE	
CRO	
CRO	
CSA	
DAL	
DAL	
DAT	
DAT	
DEB	
DEL	
DFN	
DIR	
DIS	
DIS	
DIS	
DIS	
DIS	
DIS	
DIS	
DIS	
DIS	
DOU	
DSA	
ENA	

0226 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

